NVG Scientific Sdn Bhd
Green & Scientific Computing Solutions
http://www.novaglobal.com.sg

e-Science for the masses : How Grid Computing can help your research?

Muhammad Farhan Sjaugi, S.Kom. M.Sc
Advanced System Engineer
NVG Scientific Sdn Bhd
farhansj@novaglobal.com.sg

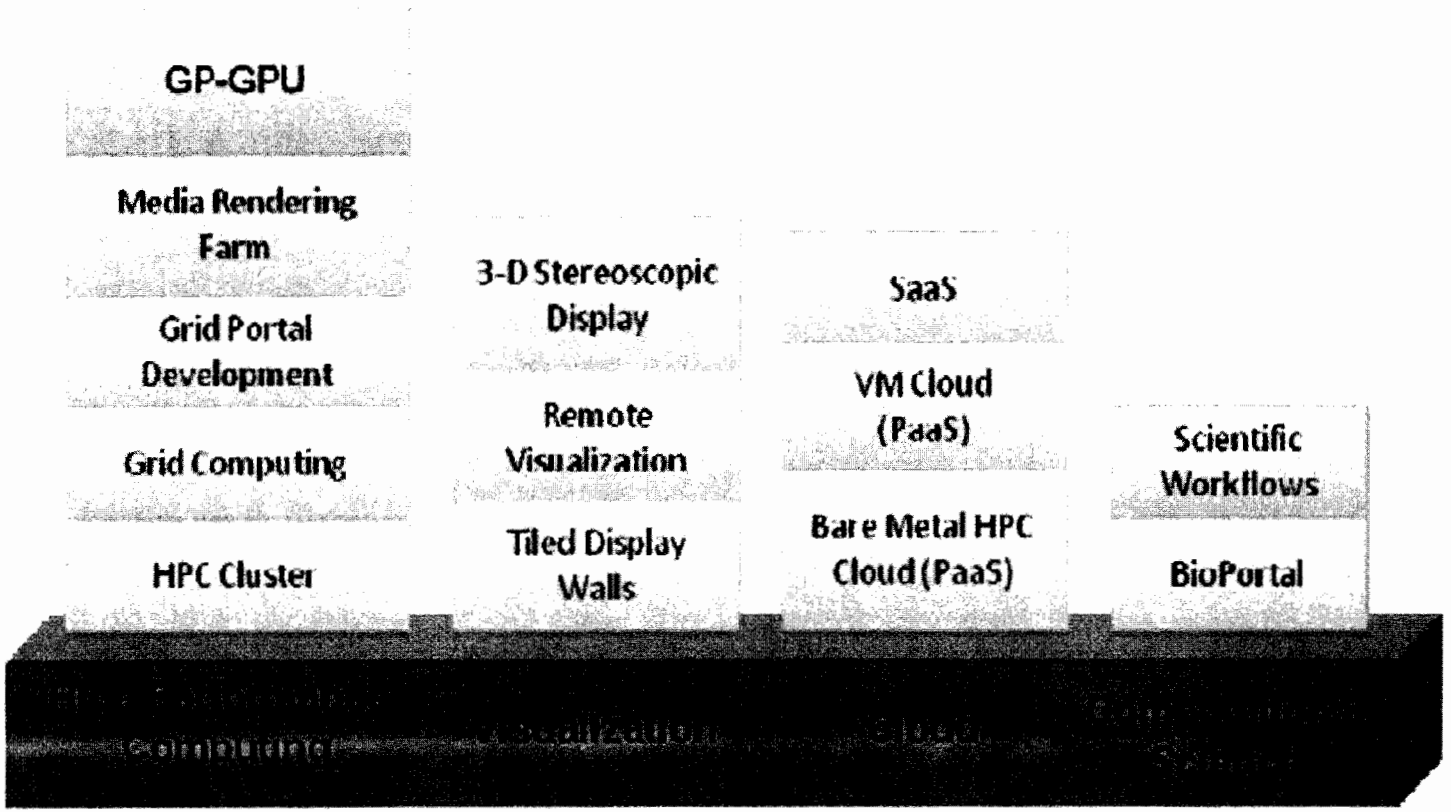
Outline

- Grid Computing Middleware
- Grid Security
- Get access to the Grid
- Getting User Certificate
- Virtual Organization Registration
- User Interface and Grid Authentication
- Job Management
- Data Management

About NovaGlobal Private Limited

- Established in 1996
- Operating in ASEAN: Singapore, Malaysia, Indonesia, Thailand and Vietnam
- In Malaysia, registered as NVG Scientific Sdn Bhd
- Platform-Independent
- Focus on UNIX/Linux HPC Cluster, Grid Computing and Windows HPC Server solutions
- Partnering with Technology Leaders

About NovaGlobal Private Limited



About NovaGlobal Private Limited

Customers

Singapore
Novartis Institute Tropical Disease
Agency of Science, Technology and Research
Eli Lilly
National University of Singapore
Nanyang Technological University
National Cancer Center Singapore
National Environmental Agency
Genome Institute of Singapore
Temasek Laboratories
Temasek Polytechnic
Republic Polytechnic

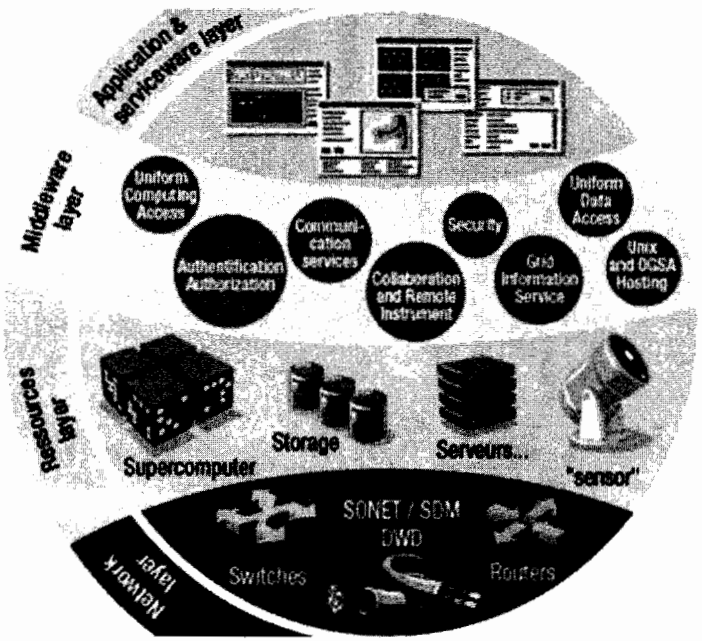
Malaysia
Multimedia Development Corporation
Universiti Teknologi Malaysia
University Malaya
University Putra Malaysia
University of Nottingham, Malaysia

Indonesia
Government Statistics

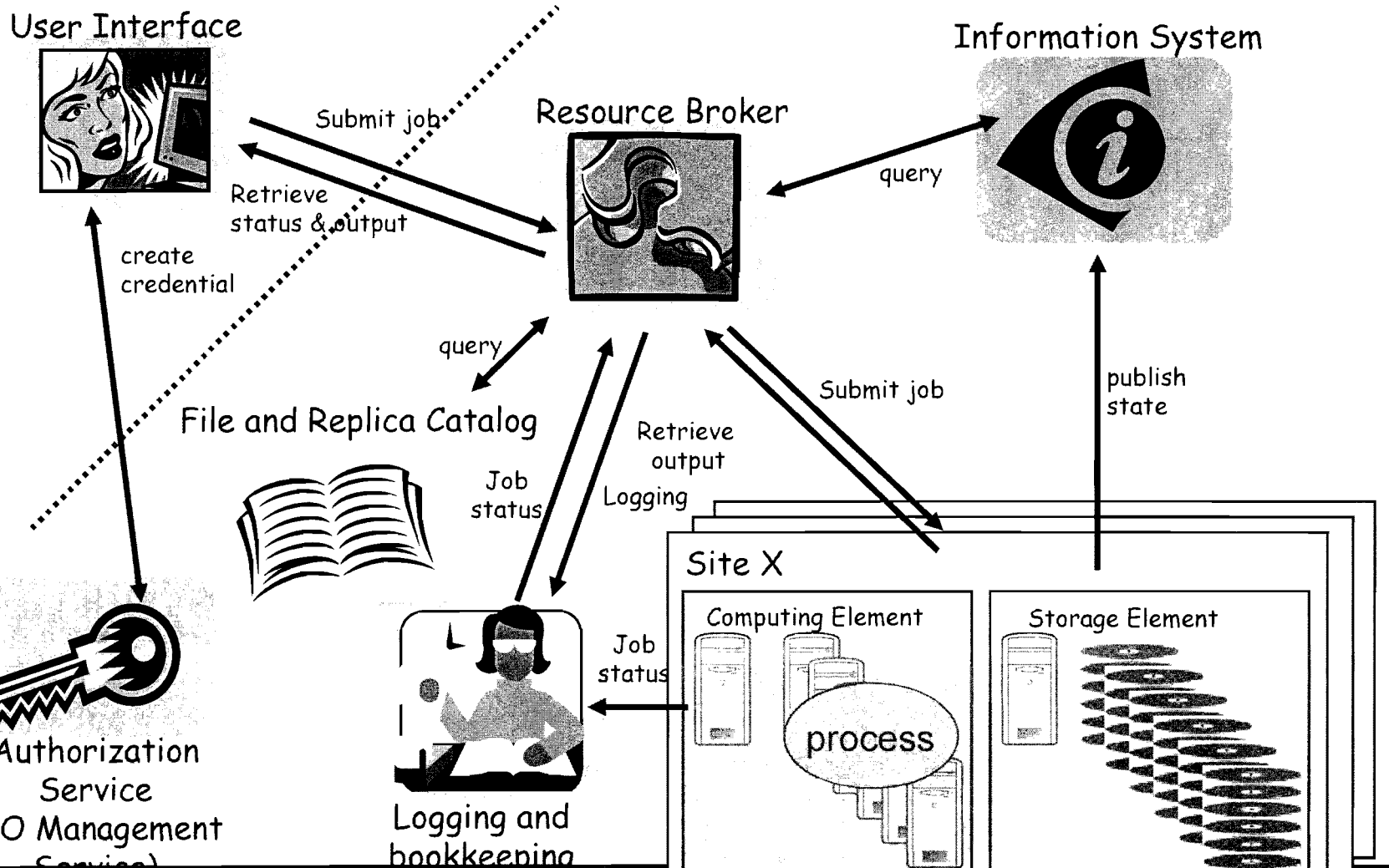
Vietnam
Government Security

Grid Computing Architecture

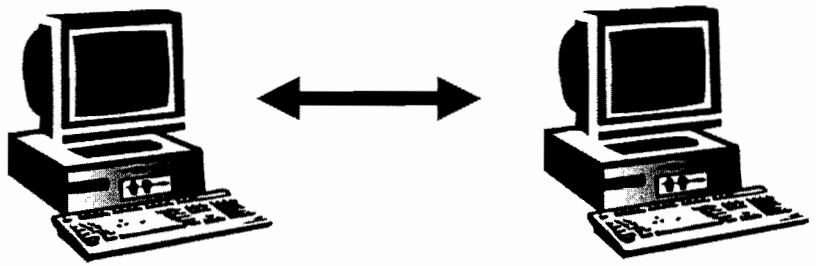
- The Grid relies on advanced software, called middleware, which interfaces between resources and the applications.



How Grid Middleware works?



Grid Security

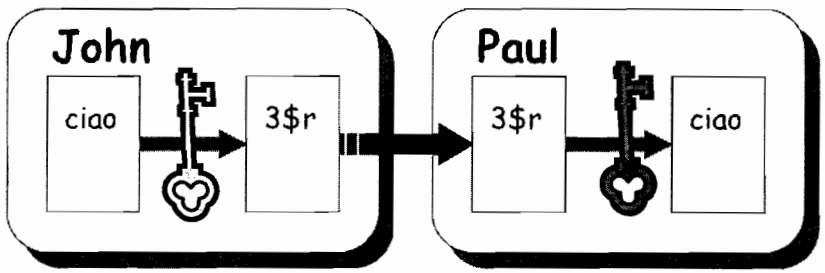


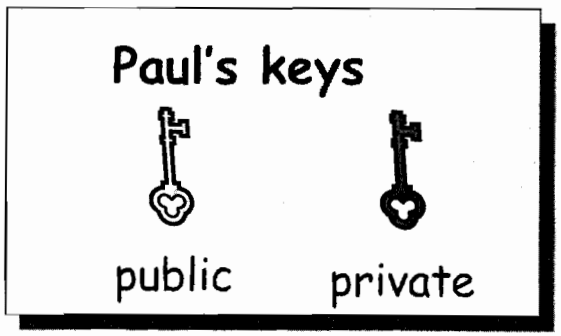
User **Grid service**

- **Authentication**
  - How can communication endpoints be identified?
- **Authorization**
  - Who is allowed to access a Grid resources
  - What are Grid members allowed to do?

Grid Security

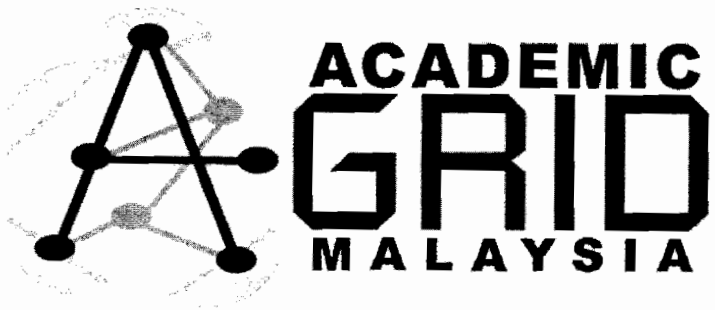
- Grid use Public Key Infrastructure (PKI) to secure the Grid resources.
- PKI Encryption at glance
  - Encryption with recipient's public key
  - Only recipient can decrypt the message

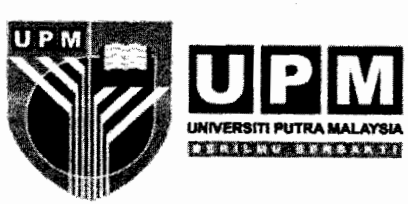




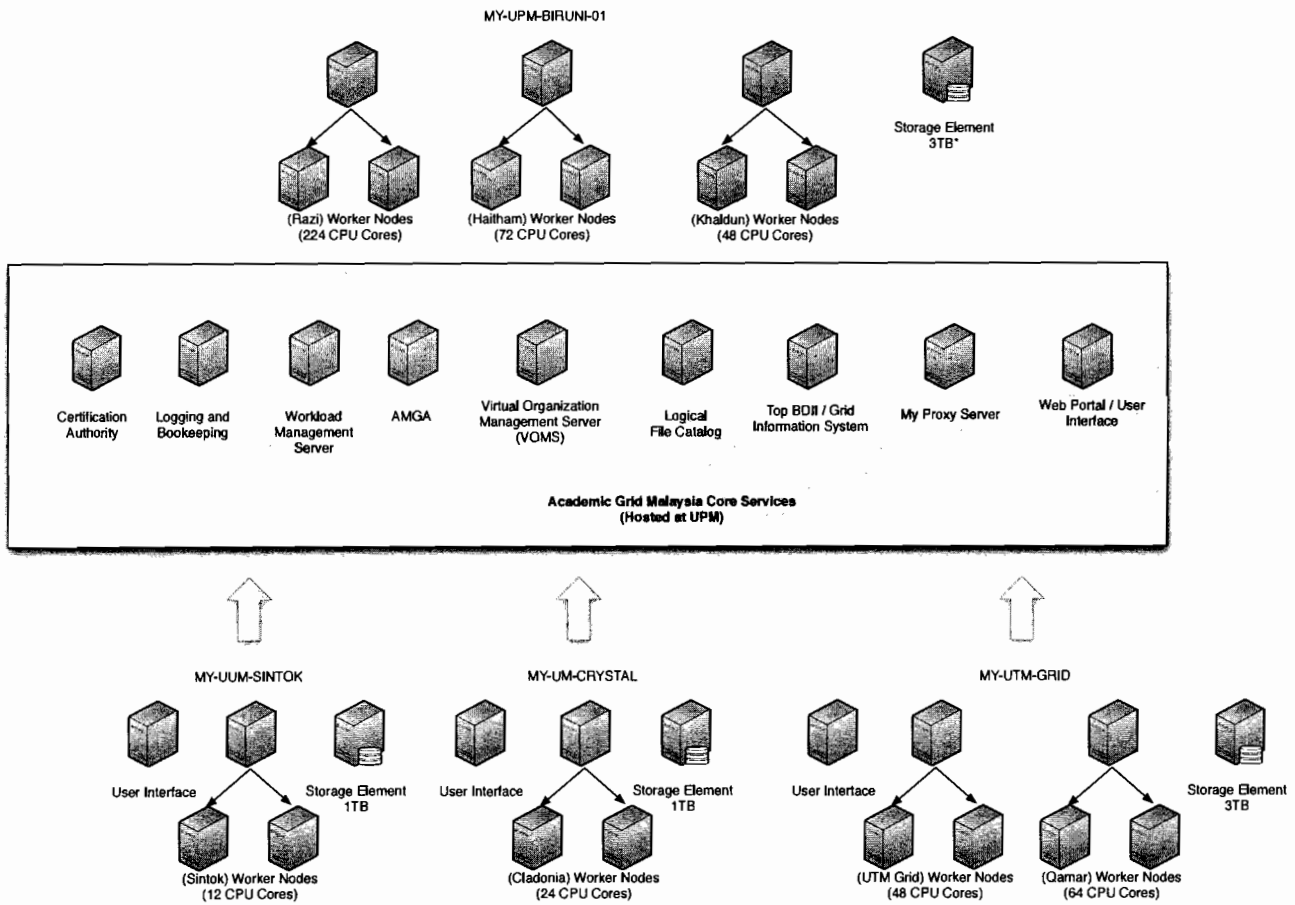
Academic Grid Malaysia

- is an open access distributed computing infrastructure Open to any Malaysian Academia from institutions and organizations that are recognized by respective ministries to be involved/ associated in/with Distributed-, Grid-, Cloud- and Emerging Computing Applications/Projects.
- Currently there are 4 resource contributors: MY-UPM-BIRUNI-01, MY-UM-CRYSTAL, MY-UTM-GRID and MY-UUM-GRID
- More than 400 core of cpus and 3TB of storage are contributed.

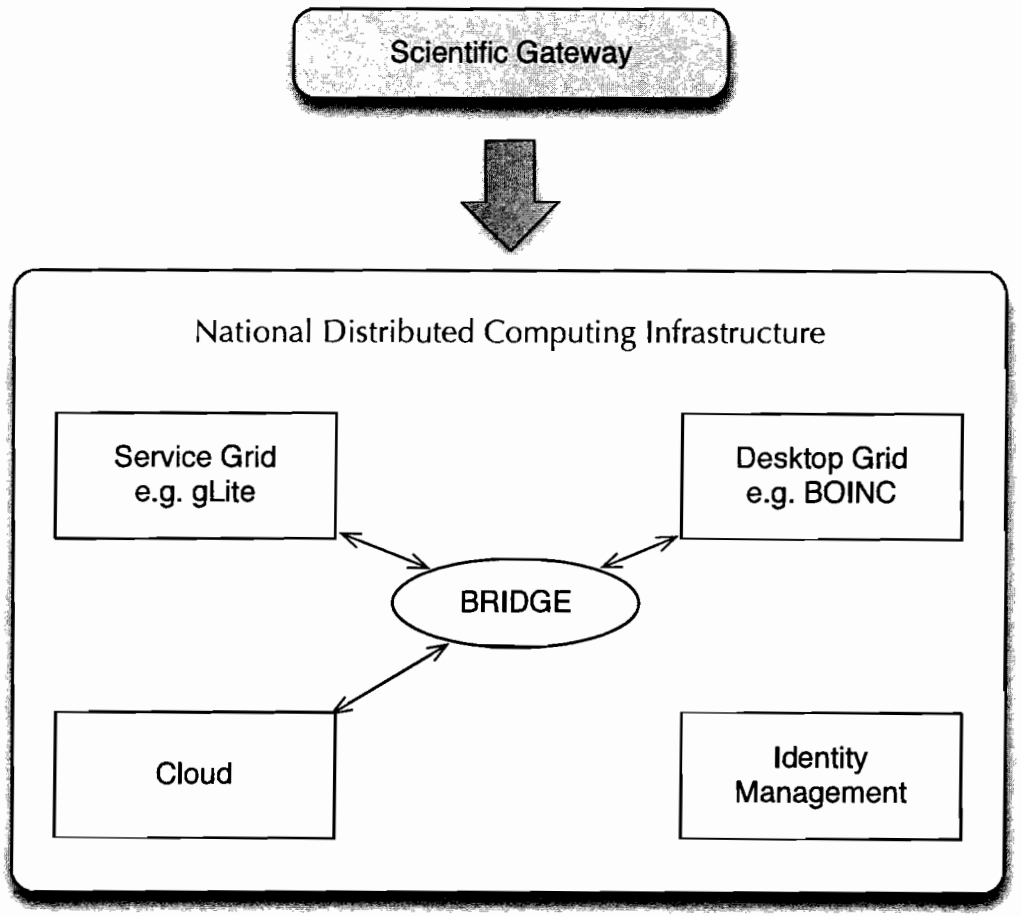




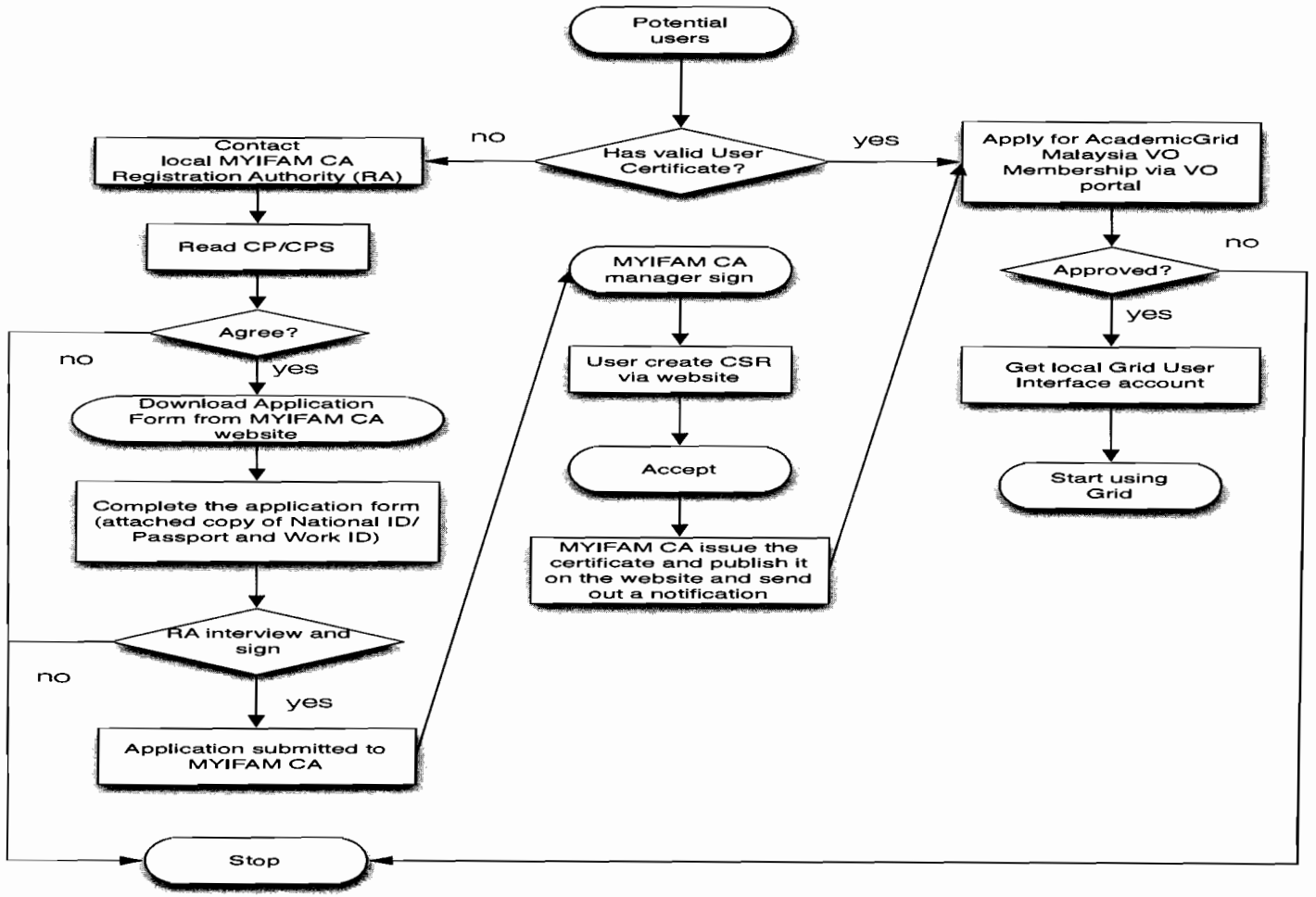
Academic Grid Malaysia



Malaysia National Distributed Computing Infrastructure Roadmap



Get access to the Grid



Getting User Certificate

- Grid user (from Malaysia) shall obtain User Certificate from MYIFAM CA.
- Please download and complete MYIFAM CA user certificate request form from:
  - http://myifam.upm.my/pub/myifam_user_cert_application_form.pdf
- Along with the application form, please attach copy of your NRIC/Passport and Staff/Student ID.
- Get your application endorsed by local Registration Authority (UUM users may get from En. Mohd Samsu Bin Sajat (mohdsamsu@uum.edu.my)).

Getting User Certificate

- Make sure you use one of these operating systems and internet browsers to apply user certificate:
  1. Windows (XP, Vista, 7)
     1. IE
     2. Mozilla Firefox (highly recommended)
  2. MAC OSX
     1. Safari
     2. Mozilla Firefox (highly recommended)
  3. Linux
     1. Mozilla Firefox (highly recommended)
- Create online CSR from:
  - https://myifam.upm.my/securearea/generate.php
- Please use your official email! Public email (e.g. gmail, yahoomail won't be accepted)
- You will be informed by MYIFAM CA when your certificate is ready.
- Your certificate is valid for 400 days and renewable.

Virtual Organization (VO) Registration

- Open Academic Grid Malaysia VO registration page at https://voms.biruni.upm.my:8443/voms/academicgrid
- You need to have your user certificate ready and installed on your internet browser before you can register.
- By default your VO membership valid for one year, but you can always extend it.

User Interface and Grid Authentication

- Login to Grid User Interface* by using any ssh client. Windows user can use putty. (For UUM users, please ask for an account from En. Mohd Samsu bin Sajat (mohdsamsu@uum.edu.my)).
- Do authentication and authorization to the grid:
  - `$voms-proxy-init --voms academicgrid`
- By default you can use the grid for 12 hours (includes time to complete your job). Shall you need more than 12 hours to complete your job, you can store a long-term proxy certificate at MyProxy server

User Interface and Grid Authentication

- To create a long term proxy certificate:
  - `$myproxy-init -d -n`
- By default your certificate stored on MyProxy server valid for 7 days (168hours). Please refer to quick reference guide shall you need less or more than default time.
- It is ok for you to login/logout from the UI without having to re-authenticate/authorize unless your proxy cert is already expired. Please refer to quick reference guide for more **voms-proxy** and **myproxy** commands.

User Interface and Grid Authentication

- To explore any computation and storage resources availability on academic grid, you can use one of these commands:
  - `$lcg-infosites --voms academicgrid ce`
  - `$lcg-infosites --voms academicgrid se`
  - `$lcg-infosites --voms academicgrid all`

Job Management

- At least 2 files are required to submit Job to the Grid; Job Description Language file (JDL) and Job Script (.sh)
- JDL file consist of information about the description and requirement in order to execute the job
- Job Script consist of instruction to run the Job

JDL Example

```
JobType          = “Normal”;
Executable       = "job1.sh";
Arguments        = "";
StdOutput        = "std.out";
StdError         = "std.err";
InputSandbox     = {"job1.sh"};
OutputSandbox    = {"std.out","std.err"};
```

Job Script Example

```
#!/bin/sh
hostname
```

Job Submission

- Submit Job:
  - `$glite-wms-job-submit -a -o jid job1.jdl`
- Check Job Status:
  - `$glite-wms-job-status -i jid`
- Retrieve Job Status
  - `$glite-wms-job-output -i jid -dir job1`

Working with Input Data

- It is possible to submit job along with some input data to the Grid.
- An example, the job script reads data from output.txt and print it to the output with inclusion of some string.
- Upon successful, it shall return output as below:

```
This is the first data
This is the second data
This is the third data
This is the fourth data
This is the fifth data
```

Working with Input Data

- Job2.jdl

```
JobType          = "Normal";
Executable       = "job2.sh";
Arguments        = "";
StdOutput        = "std.out";
StdError         = "std.err";
InputSandbox     = {"job2.sh","data.txt"};
OutputSandbox    = {"std.out","std.err"};
```

Working with Input Data

- Job2.sh

```
#!/bin/sh
for record in `cat data.txt`
do
        echo “This is the $record data”;
done
```

Working with Input Data

- Data.txt

First
Second
Third
Fourth
Fifth

Working with Arguments

- It is possible to submit job along with some arguments to the Grid.
- An example, The job script reads arguments given from the JDL file and print it to the output with inclusion of some string.
- Upon successful, the Grid will return output as below:

My name is John Doe bin Abdullah

Working with Arguments

- Job3.jdl

```
JobType          = "Normal";
Executable       = "job3.sh";
Arguments        = "John Abdullah";
StdOutput        = "std.out";
StdError         = "std.err";
InputSandbox     = {"job3.sh"};
OutputSandbox    = {"std.out","std.err"};
```

Working with Arguments

- Job3.sh

#!/bin/sh
echo "My name is **$1** Doe Bin **$2**"

- To take whole arguments as a string, use **$?**

Working with Parametric Job

- Parametric job is Job with repeated task and different value of arguments.
- Very useful for parametric study or parallel sweep applications.
- Upon successful, the Grid will returned outputs with different result according to the each parameter given.

Working with Parametric Job

- Job4.jdl

```
JobType          = "Parametric";
Executable       = "job4.sh";
StdOutput        = "std_PARAM_.out";
StdError         = "std_PARAM_.err";
Parameters       = 10;
ParameterStart   = 1;
ParameterStep    = 1;
Arguments        = "_PARAM_";
InputSandbox     = { "job4.sh"};
OutputSandbox    = {"std_PARAM_.out","std_PARAM_.err" };
```

Working with Parametric Job

- Job4.sh

```
#!/bin/sh
let "mul=10*$1"
echo "10 * $1 is equal to $mul";
```

Working with Parametric Job

- Job5.jdl

```
JobType          = "Parametric";
Executable       = "job5.sh";
StdOutput        = "std_PARAM_.out";
StdError         = "std_PARAM_.err";
Parameters       = {first,second,third};
Arguments        = "_PARAM_";
InputSandbox     = { "job5.sh"};
OutputSandbox    = {"std_PARAM_.out","std_PARAM_.err" };
```

Working with Parametric Job

- Job5.sh

```
#!/bin/sh
echo ”This is the $1 parameter";
```

Working with Program

- You can also run your simulator/program either in source code or binary format on the grid.
- However if you submit your program in source code format, there is also probability that your job will not successfully executed as the targeted cluster may not provide compiler to compile your source code.
- Here is some example of submitting c++ source code to the Grid

Working with Program

- Job6.jdl

```
JobType = ”Normal";
Executable = "job6.sh";
StdOutput = "std.out";
StdError = "std_.err”;
Arguments = ””;
InputSandbox = { "job6.sh,”hello.cpp"};
OutputSandbox = {"std.out","std.err" };
```

Working with Program

- Job6.sh

```
#!/bin/sh
g++ hello.cpp –o hello chmod +x hello
./hello
```

Working with Program

- hello.cpp

```
#include <iostream>
using namespace std;
int main(){
        cout << "Hello World!\n";
        return 0;
}
```

Working with Program

- Now we submit the program in the binary format
- Compile the program (e.g. c++ source code)

g++ hello.cpp –o hello

Working with Program

- Job7.jdl

```
JobType = ”Normal";
Executable = ”hello";
StdOutput = "std.out";
StdError = "std_.err”;
Arguments = ””;
InputSandbox = {”hello"};
OutputSandbox = {"std.out","std.err" };
```

Working with MPI Job

- In MPI-type job, we need to set filter to WMS to only assign the job to cluster that has MPI Library installed
- Currently only MPICH2, OPENMPI and OPENMPI with Infiniband are supported.
- There are two ways to submit MPI Job, either using normal job scripting or with MPI-Start wrapper also there are two ways to run the binary, either submit precompiled binary or source code

Working with MPI Job

- Job8.jdl

```
JobType         = "Normal";
CPUNumber       = 16;
Executable      = "job8.sh";
Arguments       = "";
StdOutput       = "std.out";
StdError        = "std.err";
InputSandbox    = {"job8.sh","mpi.c"};
OutputSandbox = {"std.err","std.out"};
Requirements = Member("OPENMPI",
other.GlueHostApplicationSoftwareRunTimeEnvironment);
```

Working with MPI Job

- Job8.sh

```
#!/bin/sh
export PATH=$PATH:$MPI_OPENMPI_PATH/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:
$MPI_OPENMPI_PATH/lib

mpicc mpi.c -o mpi
mpiexec mpi
```

Working with MPI Job

- mpi.c

```
#include <stdio.h>
#include <mpi.h>

int main(int argc, char *argv[]) {
int numprocs, rank, namelen;
char processor_name[MPI_MAX_PROCESSOR_NAME];

MPI_Init(&argc, &argv); MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
MPI_Comm_rank(MPI_COMM_WORLD, &rank); MPI_Get_processor_name(processor_name,
&namelen);

printf("Process %d on %s out of %d\n", rank, processor_name, numprocs);

MPI_Finalize();
}
```

Working with MPI Job

- MPI Job submission example given is the simplest MPI Job submission and supported by most of Academic Grid site (UPM, UM and UTM). However there are some limitations; The computing cluster must share home folder and use TORQUE/ PBS job as the job manager.

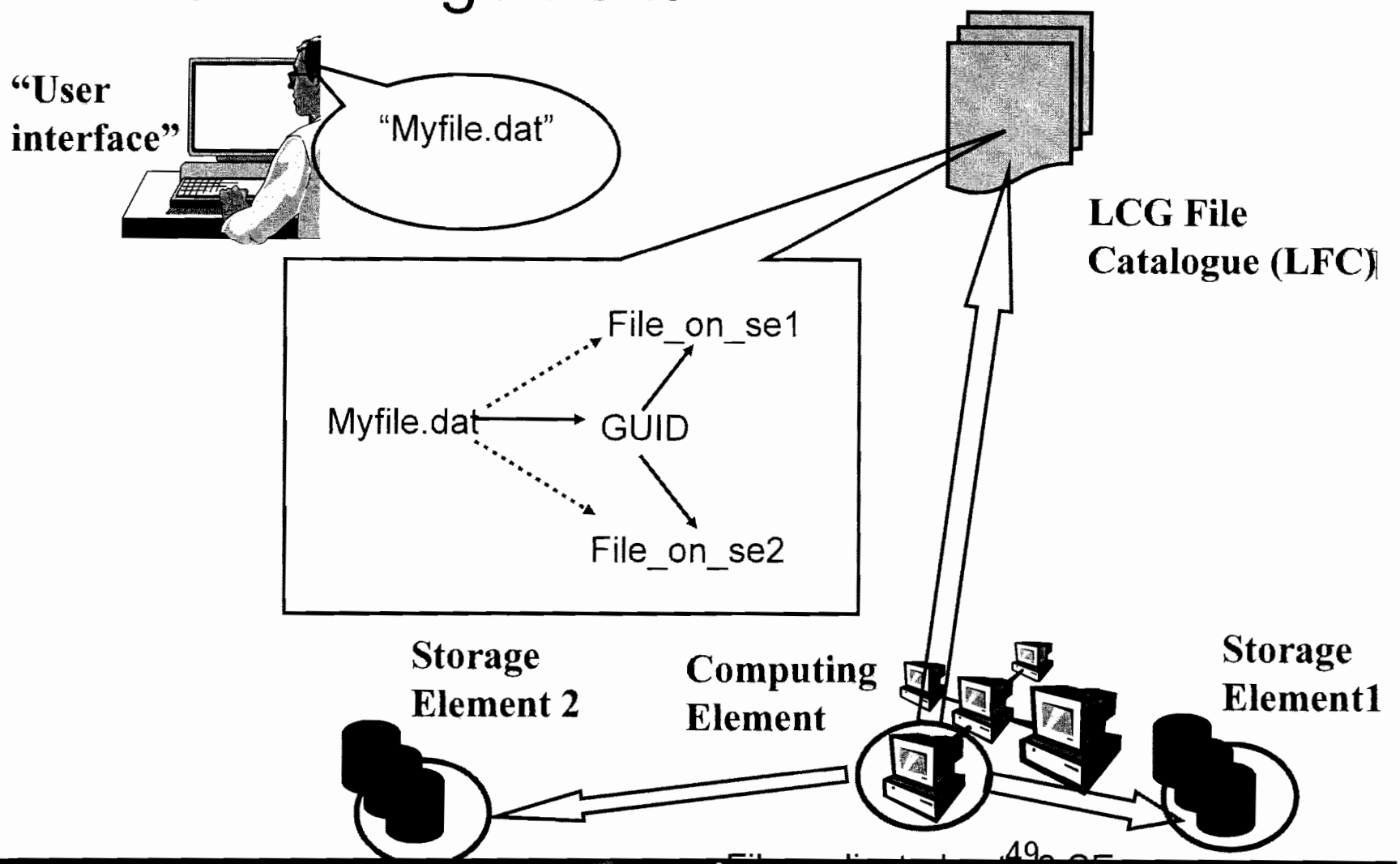
Job Submission Tips

- Before submit the Job, it is highly recommended to perform job list match to ensure that there is cluster that able to process the Job
  - $glite-wms-job-list-match –a job8.jdl
- It is highly recommended to use mpiexec command instead of mpirun
- MPI configuration in most Academic Grid site already precompiled with support for Job Manager. Hereby explicit definition of CPU numbers and Host are not required for mpiexec command

Data Management

- Simply, Grid Data Management provides all operation that all of us are used to performing
  - Uploading /downloading files
  - Creating file /directories
  - Renaming file /directories
  - Deleting file /directories
  - Moving file /directories
  - Listing directories
  - Creating symbolic links

Data Management



Data Management

- To create your home directory at File Catalog (LFC), you can use command below:
  - $lfc-mkdir /grid/academicgrid/<your directory>
- To browse file at LFC, use command below:
  - $lfc-ls /grid/academicgrid/
- Let's try uploading file to Grid Storage Element (SE) and register it at LFC:
  - $lcg-cr -l lfn:/grid/academicgrid/<your directory>/file.txt –d dpm.biruni.upm.my file:file.txt
- Now let's try to download file from SE to your local storage
  - $lcg-cp lfn:/grid/academicgrid/<your directory>/file.txt file:file2.txt
- Now let's try to delete file on SE
  - $lcg-del -a lfn:/grid/academicgrid/<your directory>/file.txt

References

- Job Description Language specification - https://edms.cern.ch/file/590869/1/WMS-JDL.pdf
- MPI-Start User Manual - https://devel.ifca.es/mpi-start/raw- attachment/version/1.0.4/mpi-start-1.0.4.pdf
- MPI on Grid - https://wiki.egi.eu/wiki/MPI_User_Guide

NVG Scientific Sdn Bhd
Green & Scientific Computing Solutions
http://www.novaglobal.com.sg

Thank You

Contact: support@novaglobal.com.sg